Available online at www.sciencedirect.com

ScienceDirect

Procedia Social and Behavioral Sciences 2 (2010) 143–152

Procedia
Social and Behavioral Sciences

Security Camera Network, Privacy Protection and Community Safety

Real-time people counting using blob descriptor

Satoshi Yoshinaga*, Atsushi Shimada, Rin-ichiro Taniguchi

Kyushu University, 744, Motooka, Nishi-ku, Fukuoka, 819-0395 Japan

Received November 13, 2009; revised December 4, 2009; accepted December 10, 2009

Abstract

We propose a system for counting the number of pedestrians in real-time. This system estimates "how many pedestrians are and where they are in video sequences" by the following procedures. First, candidate regions are segmented into blobs according to background subtraction. Second, a set of features are extracted from each blob and a neural network estimates the number of pedestrians corresponding to each set of features. To realize real-time processing, we used only simple and valid features, and the adaptive background modeling using Parzen density estimation, which realizes fast and accurate object detection in input images. We also validate the effectiveness of the proposed system by several experiments.

Keywords: Visual surveillance; people detection; people counting

1. Introduction

There are many surveillance cameras everywhere for various purposes. The surveillance images obtained from these cameras are used for security, market research, and so on. Usually, the observers monitor the surveillance images gathered from those cameras. In practice, this kind of manual monitoring has the following problems:

- Even if there are only a few observers, the surveillance cost is quite high.
- As their working hours increase, the observers' fatigue increases and, as a result, their alertness decreases.
- The surveillance cost and the observers' fatigue increase with increase of the number of cameras.

Especially, when real-time analysis of observed images is required, such as for security measures, the observers' fatigue can become a serious issue. Therefore, automatic analysis using image processing is a promising approach to such applications, and, here, we develop a system for estimating "how many pedestrians are and where they are in video sequences" in real-time.

This paper is organized as follows. In section 2 we review related work in people counting. In section 3 we introduce a people counting method based on blob features. In section 4 we show experimental details and results. Finally, we present a conclusion and future tasks in section 5.

* Corresponding author. Tel.: +81-92-802-3580; fax: +81-92-802-3579.
E-mail address: yoshinaga@limu.ait.kyushu-u.ac.jp.

1877-0428
10.1016/j.sbspro.2010.01.028

2. Related work

People counting using image processing is to estimate the number of pedestrians in input images. Information about pedestrians, including the number and the positions of them, from a people counting system based on image processing is expected to reduce the surveillance cost and the observers' fatigue. Pedestrian information can be used in a variety of potential applications. Various methods which estimate the number of people in input images have been previously proposed. They can be divided into three approaches:

1). individual pedestrian detection
2). visual feature trajectory clustering
3). feature-based regression

The first ones, which are individual pedestrian detection algorithms, estimate the number of pedestrians by detection of all of them in input images. For example, Viola et al. proposed a people counting method based on boosting appearance and motion features (Viola et al., 2005). Zhao et al. proposed a method based on Bayesian model-based segmentation (Zhao and Nevatia, 2003). However, these methods cannot be applied to very crowded scenes with significant occlusion because they need to detect and segment all pedestrians.

The second is trajectory clustering approach, in which people are counted by tracking and identifying visual features over time. The feature trajectories that exhibit coherent motion are clustered, and the number of clusters is the estimated number of pedestrians. For example, Antonini et al. proposed a people counting method in which the trajectories obtained by tracking algorithm are clustered based on their lengths and spatial locations (Antonini and Thiran, 2006). Since this approach estimates the number of pedestrians who passed within a specific time, their real-time processing is difficult.

The third approach is feature-based regression approach, which estimates the number of pedestrians by regressing the features, extracted from input image, using a regression function. These methods typically work by: 1) background subtraction; 2) extracting various features of the foreground region; and 3) estimating the number of pedestrians by a regression function of extracted feature values, e.g. liner, piece-wise liner, or neural networks. For example, Kong et al. proposed a people counting method in which neural networks as regression function is used to regress the features, such as edge orientation and blob size histograms obtained by applying background subtraction and edge detection to input image, to the number of pedestrians (Kong et al., 2005). Chan et al. proposed a method in which a Gaussian process as regression function is used to regress 28 features extracted from crowd segment obtained by the *mixture of dynamic texture* (Chan and Vasconcelos, 2005), in a privacy-preserving manner (Chan et al., 2008). However, these methods cannot estimate the positions of pedestrians in the input image, or they cannot be executed in real-time. For example, the method proposed by Kong et al. can never estimate where pedestrians are in the input image, because it extracts only one feature vector from input image. The method proposed by Chan et al. cannot realize real-time processing in principle, because the *mixture of dynamic texture* used for segmentation needs a series of successive images from the past to the future. Furthermore, it may take a lot of time to extract 28 features including complex ones, such as Minkowski dimension, homogeneity, and entropy.

Comparing the three methods, (3) feature-based regression method is the most accurate because it is able to cope with occlusion and the change of background. However, as described above, the previous methods have several remaining problems. In this paper, therefore, we propose a real-time people counting method to estimate "how many pedestrians are and where they are in video sequences".

The proposed method consists of: 1) background subtraction and shadow elimination; 2) extracting the features of the foreground blobs; and 3) estimating the number of pedestrians in each blob by neural network. We achieve the real-time processing by the following tactics:

Simplifying features: making feature extraction faster by using only simple and effective ones.

Fast segmentation: foreground objects are detected by background modeling based on fast Parzen density estimation (Tanaka et al., 2007).

3. People counting method using blob features

The outline of the proposed system is illustrated in Figure 1. The input image is segmented into blobs of moving objects, using background subtraction and shadow elimination. For each blob, various features are extracted, and they are normalized according to its approximate size in the real scene. Finally, the number of pedestrians is

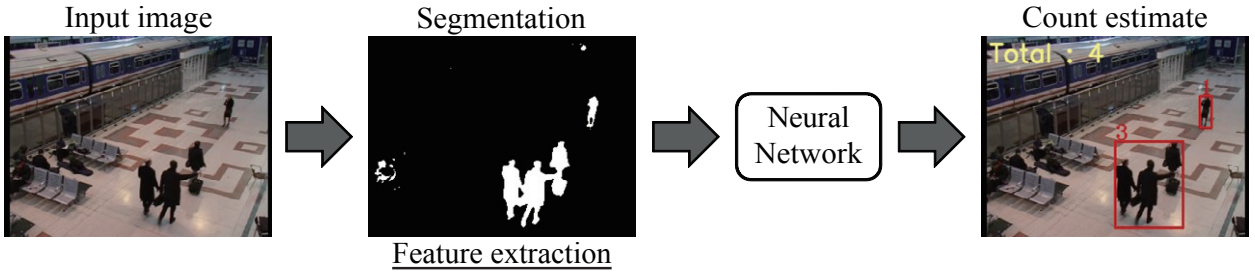


Figure 1: Outline of the people counting system

estimated in each blob by a neural network. In advance, the neural network is trained to establish the relationship between the feature values and the number of pedestrians in a blob referring to training data.

3.1. Pedestrian region segmentation

We adopt the background subtraction to detect non-background pixels in input images. We can get object regions only by subtracting the background image from an observed image without requiring prior information about the objects. In the proposed method, we adopt a fast algorithm for adaptive background model construction using Parzen density estimation (Tanaka et al., 2007) to detect object regions quickly and accurately without suffering from influence of small intensity or color changes such as illumination changes. This algorithm estimates background model from pixel values observed in video sequence using Parzen density estimation, which is non-parametric density estimation. However, shadow regions cast by objects, whose size and position changes with time, are detected as objects by the background subtraction, as shown in Figure 2(b). Because shadow regions may affect the people counting system, we need to remove them from foreground obtained by background subtraction. Therefore, we adopt a shadow detection method using YUV color (Schreer et al., 2002) to eliminate shadow regions. This method is based on the observation that shadows cast on a surface will reduce the values of the intensity and the saturation by similar small ratio, and that the hue value of the shadows are similar to the original background. In other words, this method is based on the observation that shadows cast on a surface will equally attenuate the values of three component of its YUV color. Finally, we can detect only the object regions which does not include shadow regions by using background subtraction followed by shadow elimination as described above (see Figure 2(c)).

(a) Input image (b) Subtracting the background (c) Eliminating the shadow regions

Figure 2: Pedestrian region segmentation

3.2. Feature extraction

In practice, the relationships between the features and the number of pedestrians contained in the blob are usually non-linear, due to occlusion, segmentation errors, the individual difference of pedestrian (e.g. their height) and so on. Therefore, to model these non-linearities, we have adopted neural-network-based approach using 6 features extracted from each blob of the detected foreground regions.

Blob features: These features capture the shape and the size of a blob segmented as foreground.

- **Area** – the total number of pixels in the blob.
- **Perimeter length** – the total number of pixels on the blob perimeter.
- **Perimeter-area ratio** – the ratio between the blob perimeter and area, which measures the complexity of the blob shape. Blobs of high ratio contain bumps in their perimeter, which may be indicative of the number of people contained within.
- **Total edge length** – the total number of edge pixels contained in the blob. The edges contained in the blob are a strong clue about the number of people in the blob. A Canny edge detector is applied to the entire image, the edge image is masked by the segmented regions, and the edge length is computed.

Rectangle features: These features capture the ratio and the position of the rectangle containing blob segmented as foreground.

- **Aspect ratio** – the ratio between height and width of the rectangle, which measures pedestrian configuration in the blob.
- **Protrusion status** – the status associated with whether entire bodies of pedestrians in the blob are inside of an input image or not. When the upper side of the rectangle containing blob coincides with the upper side of the input image, we judge that some parts of pedestrian upper bodies are not inside the image, and the status is set to be "no upper body." The other sides are dealt with in the same way. When a part of the body is not inside of the image, the extracted image feature values are changed, and the protrusion status is introduced in order to deal with such situations. The relation between the protrusion status and the image features are also trained in the neural network.

3.3. Features normalization

We have to consider the effects of perspective projection of the camera system, because, in most situations, the plane of camera projection is not parallel to the ground plane, where pedestrians move. Therefore, the area of the same object varies with its position on the input image, and the features extracted from the blob also vary with the scale of the object. To deal with this problem, we have investigated the effectiveness of feature normalization, and we have adopted two approaches: explicit normalization and implicit normalization.

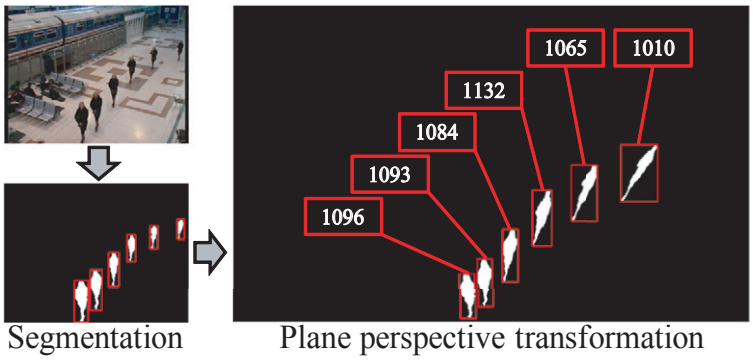


Figure 3: Example of segmentation and a plane perspective transformation associated with same person

Explicit normalization: In this method, we first transform an input image so that the size of pedestrian region is independent of its position by a plane perspective transformation called *homography*. In the plane perspective

transformation, the correspondence between a point (u,v) on the input image and a point (x,y) the ground plane is described in a 3×3 matrix H as follows:

$$u(h_{31}x+h_{32}y+h_{33})=h_{11}x+h_{12}y+h_{13},$$
$$v(h_{31}x+h_{32}y+h_{33})=h_{21}x+h_{22}y+h_{23}.$$

Four point correspondences are sufficient to solve these equations for H. We can transform input image into the image, in which the area of a certain object/person at some point is approximately equal to area of the same object/person at another point, as shown in Figure 3. However, pedestrians lean to the left or right in transformed image, because the pedestrians are not coplanar with the ground plane on which they are standing vertically. In this case, the features, such as blob perimeter or edge, still depend on the point on the transformed image. Therefore, we normalize features using a ratio (S_a/S_b) between the area (S_a) transformed by a plane perspective transformation and former area (S_b) of the blob. For features based on the area (e.g. blob area), the ratio (S_a/S_b) is applied to them. For linear features (e.g. blob perimeter or edge), the square-root of the ratio is $\sqrt{(S_a/S_b)}$ used. Perimeter-area ratio is computed by the blob area and the blob perimeter is normalized in this way. The features based on the rectangle containing blob are extracted from former input image, because they are independent of their scales, i.e., independent of their positions.

Implicit normalization: In this method, we extract the 7 features from each blob: the 6 features as described in section 3.2 and the position parameter, and we normalize the features implicitly by training a neural network with training data including the scale parameter. Since the area of the same object varies with its position on the input image, we can use the positional information of blob to implicitly normalize feature values. Then, the coordinate values of the rectangle containing blob is used as positional information of blob. Thus, we can make a neural network estimate scale transition of pedestrian on the image with the number of pedestrians contained in the blob.

3.4. Estimation of the number of pedestrians by a neural network

To capture non-linear relationship between the features and the number of pedestrians contained in the blob from training data, we use a neural network. In our system, the neural network model has two hidden layers, as shown in Figure 4. The input layer and two hidden layer have *n* units, which correspond to the dimension of features. Thus, *n* is either 6 or 7, depending on which normalization is used. The output layer has only one unit, which correspond to the number of pedestrians contained in blob. Sigmoid activation function is used for all the nodes in the network. Then the network are trained with Resilient back propagation (RPROP) algorithm (Riedmiller and Braun, 1993).

To make training data, we have used blobs detected automatically and the features extracted from the blobs. The number of pedestrians in each of the blobs is given manually. After the neural network has trained with this training data, we can use the network to estimate the number of pedestrians online.

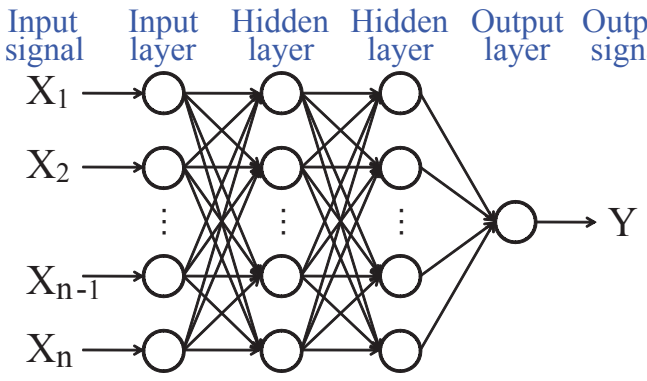


Figure 4: The neural network used in proposed method

4. Experiment

In this section, we describe the results of following experiments: 1) basic evaluation about the accuracy, 2) computation cost evaluation, and 3) further verification with additional dataset.

4.1. Basic evaluation about the accuracy

In this experiment, we have used data set of PETS2006 (PETS2006) (see Figure 1 and Figure 2) after the image resolution is reduced into 320 × 240 pixels. This data set consists of 3020 image frames, and we used the images from the 2035th to 3020th frame for training the neural network. Then, selected a test frame every 7th image frame in the data set and we used first 290 test frames here, which do not overlap with the training frames, for evaluations of the accuracy and error.

In the Table 1, we evaluated how correctly our proposed method could estimate the number of pedestrians contained in a blob, e.g., how many blobs containing one pedestrian were recognized as "one" and were incorrectly recognized as "two". In the cases of the blobs containing fewer than two pedestrians, Table 1 shows that the proposed method could recognize the number of pedestrians contained in them correctly. This is because the neural network could learn various information about such blobs, because they were observed continually in the training frames. On the other hand, Table 1 also shows that the proposed method could not estimate the number of pedestrians contained in the blobs correctly, which contained three pedestrians. This is because the neural network could learn little information about such blobs, which were rarely observed in the training frames.

Table 1: Confusion matrix of the number of pedestrians contained in a blob (PETS2006)

(a) Explicit normalization

		Estimated value					Total	Accuracy (%)
		0	1	2	3	4		
Ground truth	0	3399	37	0	0	0	3436	98.9
	1	41	851	36	0	0	928	91.7
	2	0	38	94	0	0	132	71.2
	3	0	0	5	10	4	19	52.6

(b) Implicit normalization

		Estimated value					Total	Accuracy (%)
		0	1	2	3	4		
Ground truth	0	3388	48	0	0	0	3436	98.6
	1	53	741	134	0	0	928	79.8
	2	0	23	108	1	0	132	81.8
	3	0	0	2	5	8	19	26.3

In the Table 2, we have evaluated the accuracy obtained by comparing the ground truth with the values estimated by the proposed method with two different normalization described in section 3.3, i.e., explicit normalization and implicit normalization. Figure 5 shows the result of people counting, where the shaded ranges are frames used for training. In the Table 2, False-Positive and False-Negative are defined respectively as follows:

False-Positive – the number of errors caused by counting non-pedestrian objects as pedestrian and caused by counting the number of pedestrians more than the value of ground truth.

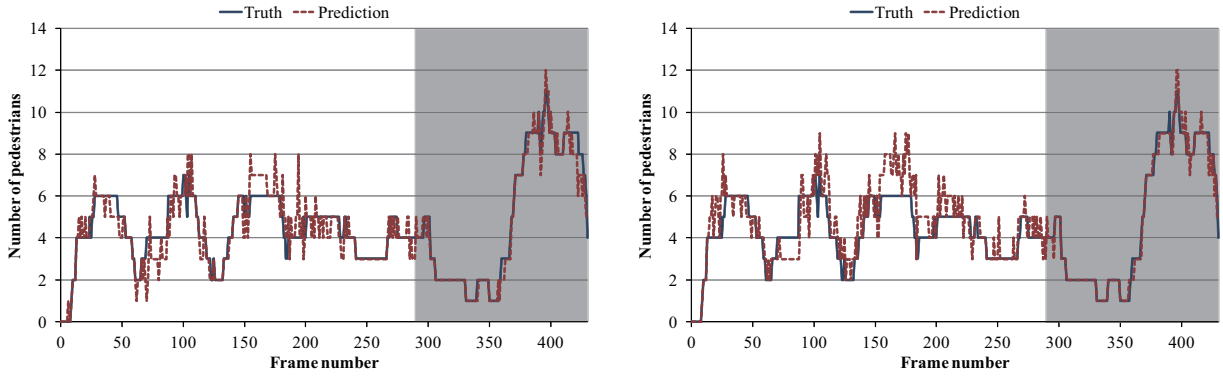
False-Negative – the number of errors caused by missing pedestrians and caused by counting the number of pedestrians less than the value of ground truth.

Table 2 shows that there is not much difference in the False-Negative value regardless of which normalization method is used. On the other hand, Table 2 also shows that number of False-positives in the implicit normalization is considerably larger than that in the explicit normalization. It is also indicated that the number of errors which are

caused by incorrectly recognizing blobs containing one pedestrian as "two pedestrians" were much larger in the implicit normalization than in the explicit normalization. In the implicit normalization, the image features supplied to the neural network are position dependent, but the neural network cannot acquire the position dependency of the image features very accurately. This is partly because in the implicit normalization the image features are calculated in small blobs in the distant area, and the precision of the feature values is not so high as those extracted in the explicit normalization.

Table 2: The accuracy of proposed method

	Explicit normalization	Implicit normalization
Confirmed Pedestrians	1249	1249
False-Positive	77	199
False-Negative	84	78
Total-Error	161	277
Accuracy (%)	87.1	77.8



(a) Explicit normalization

(b) Implicit normalization

Figure 5: People counting results

4.2. Computation cost evaluation

We have evaluated the computation time to process one image frame. We have used a PC with a Pentium IV 3.2GHz and 2GB memory. Table 3 shows the processing speed of the proposed method. Here, Segmentation, Pedestrian Estimation, and Total are defined respectively as follows:

Segmentation – time required for object segmentation by adaptive background modeling.

Pedestrian Estimation – time required for estimating the total number of pedestrians in the object segmentation result.

Total – total time including segmentation and pedestrian estimation.

Table 3: Computation time to process one image frame

	Explicit normalization	Implicit normalization
Segmentation(ms)	49.1	49.1
Pedestrian Estimation(ms)	39.2	3.9
Total(ms)	88.3	53.0

Table 3 shows that the proposed method is executed faster than 10fps, regardless of whether normalization is employed, and this is efficient enough. Pedestrian estimation with the implicit normalization is quite fast, because the position dependency of the image features is incorporated into the neural network and because no additional image processing is required at all. On the contrary, the explicit normalization requires a plane perspective transformation in every frame. However, from the viewpoint of accuracy, we should conclude that the explicit normalization is preferable.

Figure 6: Example of data set of UCSD

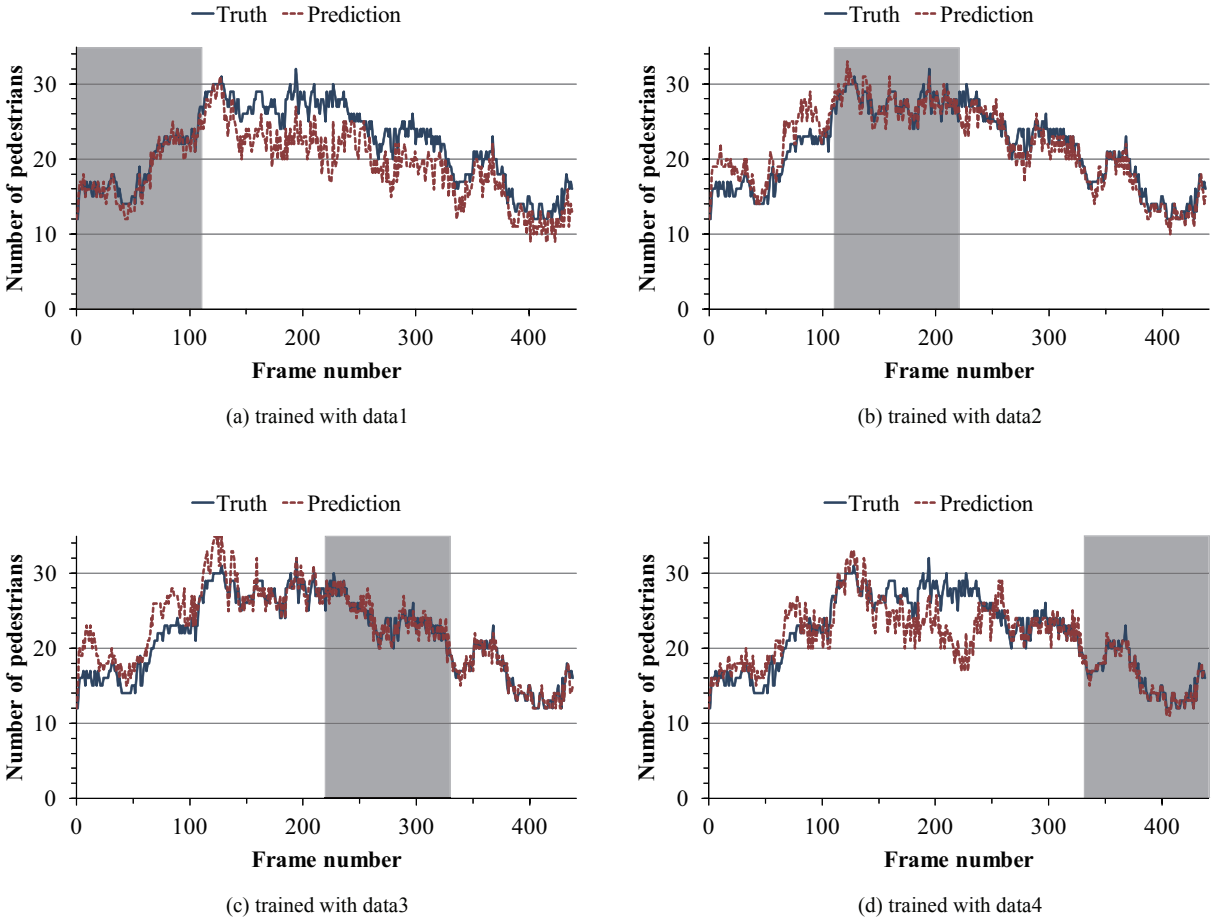
4.3. Further verification with additional dataset

According to section 4.1, it was found out that the proposed method with the explicit normalization is more accurate than that with the implicit normalization. Hence, we validate the effectiveness of proposed method with the explicit normalization using another data set, UCSD (University of California, San Diego) data set. The resolution is also reduced into 320 x 240 pixels. UCSD data set is acquired by observing wider area than that of PETS2006 used in section 4.1 and captures many pedestrians in the scene. Figure 6 shows example of data set of UCSD. We used 440 image frames in this data set to make training data for cross-validation. After we divided this training data into four parts (data1;1–110, data2;111–220, data3;221–330, and data4;331–440), we validated the proposed system using them by cross-validation. Table 4 shows the accuracies corresponding to training parts. We used the image frames except for training data to evaluate the accuracy and error. The results of people counting are shown in Figure 7, where the shaded ranges are frames used for training.

Table 4: Cross-validation for data set of UCSD

	Training data used			
	data1	data2	data3	data4
Confirmed Pedestrians	7492	6465	6860	7734
False-Positive	108	412	592	522
False-Negative	1216	365	207	809
Total-Error	1324	777	799	1331
Accuracy	82.3	88.0	88.4	82.8

Table 4 shows that the accuracy of proposed method is more than 80%, regardless of which training data is used, which shows the effectiveness of the proposed method. Precisely speaking, the accuracy becomes a bit lower when data1 or data4 is used for training. Since big groups of pedestrians were rarely observed in data1 and data4, the neural network could not learn enough information related to them. Therefore, the estimation accuracy became lower. On the other hand, the data2 and data3 included such big group data. This resulted in a better result.



(a) trained with data1

(b) trained with data2

(c) trained with data3

(d) trained with data4

Figure 7: People counting result used each training data

Table 5: Confusion matrix of the number of pedestrians contained in a blob (data3: UCSD)

		Estimated value															Total	Accuracy (%)
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14		
Ground truth	0	276	106	0	0	0	0	0	0	0	0	0	0	0	0	0	382	72.3
	1	18	2659	149	2	0	0	0	0	0	0	0	0	0	0	0	2828	94.0
	2	0	108	462	142	5	0	0	0	0	0	0	0	0	0	0	717	64.4
	3	0	1	19	232	34	4	4	0	0	0	0	0	0	0	0	294	78.9
	4	0	0	0	19	92	8	2	0	0	0	0	0	0	0	0	121	76.0
	5	0	0	0	0	20	27	14	2	0	0	0	0	0	0	0	63	42.9
	6	0	0	0	0	0	8	11	2	4	3	1	0	0	0	0	29	37.9
	7	0	0	0	0	0	1	1	7	10	8	6	1	0	0	0	34	20.6
	8	0	0	0	0	0	0	0	0	7	5	2	1	0	0	0	15	46.7
	9	0	0	0	0	0	0	0	0	4	2	4	2	1	0	0	13	15.4
	10	0	0	0	0	0	0	0	0	0	1	5	3	0	0	0	9	55.6
	11	0	0	0	0	0	0	0	0	0	0	2	5	0	0	0	7	71.4
	12	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	4	100
	13	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0	3	33.3
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

In the Table 5, we evaluated recognition rates based on the difference between the numbers of pedestrians contained in a blob, exactly as with the Table 1 in the section 4.1, using data3 which provided the best result in the cross-validation. In the cases of the blobs containing fewer than four pedestrians, Table 5 shows that the proposed method could recognize the number of pedestrians correctly. On the other hand, in the cases of the blobs containing more than five pedestrians, the proposed method could not estimate the number of pedestrians very correctly, because such blobs were rarely observed in the training frame and the neural network could learn little information about them. However, Table 5 shows that most of the differences between the ground truth and the estimated value were within "two", in the cases of the blobs which rarely observed in the training frames. Therefore, the proposed method will be able to estimate the number of pedestrians more correctly, if we have more training data, especially for larger blobs.

5. Conclusion

In this paper, we have proposed a method which estimates "how many pedestrians are and where they are in video sequences" in real-time. We have shown that the proposed method is executed faster than 10fps and its accuracy is higher than 80%.

There are remaining works as follows:

Improvement of background subtraction: In the proposed method, background subtraction is used for segmentation. For background subtraction, we have used the background model using Parzen density estimation. Because this model is based on pixel values observed in a sequence of the latest N frames, pedestrians staying at the same place are incorporated into the background model and they cannot be correctly extracted. Therefore, in those cases, a neural network tends to output incorrect estimation, such as the wrong number of pedestrians and incorrect labeling as non-pedestrian regions. This problem will be solved by adaptive modeling of detected blobs, where the background model around stationary blobs is not updated.

Research of better features: There might be better features than those used in the proposed method. Hence, we have to try various combination of features to get better performance.

References

Antonini, G.& Thiran, J. P. (2006). Counting Pedestrians in Video Sequences Using Trajectory Clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, 16, 1008–1020.

Chan, A. B., Liang, Z.-S. J., & Vasconcelos, N. (2008). Privacy Preserving Crowd Monitoring: Counting People without People Models or Tracking. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Chan, A. B., & Vasconcelos, N. (2005). Mixtures of Dynamic Textures. *IEEE International Conference on Computer Vision.*

Kong, D., Gray, D., & Tao, H. (2005). Counting pedestrians in crowds using viewpoint invariant training. *British Machine Vision Conf.*

Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006). In Conjunction with IEEE Conference on Computer Vision and Pattern Recognition 2006. URL: http://ftp.pets.rdg.ac.uk/PETS2006/, (accessed 2008/01/15).

Riedmiller, M. & Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. *IEEE International Conference on Neural Networks.*

Schreer, O., Feldmann, I., Golz, U., & Kauff, P. (2002). FAST AND ROBUST SHADOW DETECTION IN VIDEOCONFERENCE APPLICATION. *4th IEEE Intern. Symposium on Video Proces. and Multimedia Comm*, 371–375.

Tanaka, T., Shimada, A., Arita, D., & Taniguchi, R. (2007). A Fast Algorithm for Adaptive Background Model Construction Using Parzen Density Estimation. *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).*

Viola, P., Jones, M., & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision (IJCV)*, 63(2), 153–161.

Zhao, T. & Nevatia, R. (2003). Bayesian human segmentation in crowded situations. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2, 495–466.